INCORPORATING PHYSICAL KNOWLEDGE ABOUT THE FORMATION OF NITRIC OXIDES INTO EVOLUTIONARY SYSTEM IDENTIFICATION

Stephan Winkler[a], Markus Hirsch[b], Michael Affenzeller[c], Luigi del Re[d], Stefan Wagner[e]

[a], [c], [e] Heuristic and Evolutionary Algorithms Laboratory
School of Informatics, Communications and Media; Upper Austria University of Applied Sciences
Softwarepark 11, 4232 Hagenberg, Austria

[b] Linz Center of Mechatronics
Altenbergerstraße 69, 4040 Linz, Austria

[d] Institute for Design and Control of Mechatronical Systems
Johannes Kepler University Linz
Altenbergerstraße 69, 4040 Linz, Austria

[a] stephan.winkler@heuristiclab.com, [b] markus.hirsch@lcm.at,
[c] michael.affenzeller@heuristiclab.com, [d] luigi.delre@jku.at, [e] stefan.wagner@heuristiclab.com

ABSTRACT

Genetic programming (GP) is an evolutionary optimization method that has already been used successfully for solving data mining problems in the context of several scientific domains. For example, the identification of models describing the nitric oxides (NO_x) emissions of diesel engines has been investigated intensively, very promising results were obtained using GP.

In the standard GP process, all model structures (as well as parameter settings) of models are created during an evolutionary process; populations of models are evolved using the genetic operators crossover, mutation and selection. In this paper we discuss several possibilities how a priori knowledge can be integrated into the GP process; we have used physical knowledge about the formation of NO_x emissions in a BMW diesel engine, test results are given in the empirical tests section.

Keywords: system identification, genetic programming, incorporation of physical knowledge

1. GENETIC PROGRAMMING

Genetic programming (GP) is an optimization technique based on the theory of genetic algorithms, designed for automatically creating programs that solve a given problem situation. A genetic algorithm (GA) works with a set of candidate solutions called population; during the execution of the algorithm each individual has to be evaluated, which means that a value indicating the fitness is returned by a fitness function. New individuals are created on the one hand by combining the genetic make-up of two parent solution candidates producing a new child, and on the other hand by mutating some individuals, i.e. changing randomly chosen parts of genetic information. Beside crossover and mutation, the third decisive aspect of genetic algorithms is selection: Usually, the individual's probability to propagate its genetic information to the next generation is relative to its fitness; the better a solution candidate's fitness value, the higher the probability, that its genetic information will be included in the next generation's population. This procedure of crossover, mutation and selection is repeated over many generations until some termination criterion is fulfilled.

Similar to GAs, GP works by imitating aspects of natural evolution to generate a solution that maximizes (or minimizes) some fitness function; a population of solution candidates evolves through many generations towards a solution using evolutionary operators; the main difference is that, whereas GAs are intended to find an array of characters or integers representing the solution of a given problem, the goal of a GP process is to produce a computer program solving the problem at hand. Typically the population of a GP algorithm contains a few hundreds or even thousands of individuals and evolves through the action of operators known as crossover, mutation and selection. Figure 1 visualizes how the GP cycle works (Langdon and Poli 2002): As in every evolutionary process, new individuals are created and tested, and the fitter ones in the population succeed in creating children of their own; unfit ones die and are removed from the population.

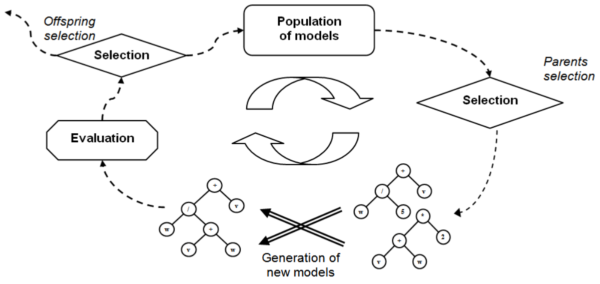


Figure 1: The extended genetic programming cycle.

2. GP BASED SYSTEM IDENTIFICATION

In general, the process of building dynamic mathematical models from measured data is called system identification (Ljung 1999). In this context, a dynamical model is a mathematical description of the dynamic behavior of a system or process which is to be identified.

The main goal here is to determine the relationship of a dependent (target) variable t to a set of specified independent (input) variables x. Thus, what we want to get is a function f that uses x and a set of coefficients w such that

$$t = f(x, w) + \varepsilon \qquad (1)$$

where ε represents the error (noise) term. Applying this procedure we assume that a model can be created with which it will also be possible to predict correct outputs for other data examples (test sample); from the training data we want to generalize to situations not known (or allowed to analyze) during the training phase.

Genetic programming has been repeatedly used successfully for building formulas that describe the behavior of systems from measured data, see for example (Koza 1992; Kejzer and Babovic 1999; Langdon and Poli 2002; Alberer, del Re, Winkler, and Langthaler 2005; del Re, Langthaler, Furtmüller, Winkler, and Affenzeller 2005; Winkler, Affenzeller, and Wagner 2006; Winkler, Affenzeller, and Wagner 2007). When it comes to evaluating a model *m* (a solution candidate in a GP based modeling algorithm, e.g.), the formula has to be evaluated on a certain set of evaluation (training) data *X* yielding the estimated values *E(m,X)*. These estimated target values are compared to the original values *T*, i.e. those which are known from data retrieval (experiments) or calculated applying the original formula to *X*. This comparison is done calculating the error between original and calculated target values; there are several ways how to measure this error, one of the simplest and probably most frequently used one being the mean squared error function.

Since 2002 we have been developing a GP based structure identification framework which also used these further developed selection principles. The HeuristicLab (Wagner and Affenzeller 2005), a framework for prototyping and analyzing optimization techniques for which both generic concepts of evolutionary algorithms and many functions to evaluate and analyze them are available, is the basis for this implementation.

3. INCLUDING A PRIORI KNOWLEDGE INTO GENETIC PROGRAMMING BASED SYSTEM IDENTIFICATION

A lot of research work has already been done regarding the use of already existing knowledge and known constraints in evolutionary system identification. Keijzer and Babovic (Keijzer and Babovic 1999), for example, describe the design of dimensionally aware GP; here, the fact that physical measurements are generally accompanied by their units of measurement is utilized leading to an extension of GP that considers the information given by the units of measurement.

In many modeling problem situations there is at least partial knowledge available about the system's structure. If the whole structure was known, then we would not necessarily need a structural system identification method as GP; but, as already insinuated, we often only know something about a certain part of the system at hand, but not the total system's structure; examples are shown in Figure 2.

Three possibilities how a priori knowledge can be incorporated into genetic programming based system identification are to be described in the following:

1. *Introduction of synthetic variables*: The most simple way to handle case 1 is to introduce an additional variable into the data base; this new variable's values are calculated according to the subsystem's model. The modeling process is thus able to incorporate this synthetic variable into models for the total system. This procedure is of course applicable and frequently used for any modeling approach. Still, subsystems as described in modeling case 2 cannot be handled using this approach since not all inputs for the modeled unit are known.
2. *Seeding parts of the population*: A genetic programming specific possibility to handle case 2 is to model the known part of the subsystem as a GP model (formula) m and to inject it into the population intentionally. This injection can be done during the population initialization phase as well as in any other phase of the GP process; in any case a certain number of individuals in the population or of the models created by crossover or mutation has to be replaced by m. For the particular example given in the right part (b) of Figure 2 this model could be for example /(+(X1, X3, 0), 1); the rest of m's inputs has to be modeled by the evolutionary process, the placeholder terminals 0 and 1 should then be replaced by appropriate subsystem representations. Furthermore, this partial model can be (by crossover) inserted into other models and so become a part of the total system's model. This model is shown in the left part (a) of Figure 2.
   Still, this approach comes with two major drawbacks: On the one hand, the models inserted into the population could be assigned very low fitness values and might thus be eliminated out of the population immediately. On the other hand, the models inserted into the population could be assigned very high fitness values, especially when the core of the system is modeled very accurately. The problem here is that these super-individuals could be so dominant in comparison to all other models, and this could have the effect that the population

immediately converges to a local optimum so that premature convergence could happen.

3. *Introduction of particular definitions in the functional base:* The most flexible possibility is surely to introduce particular functions and terminals with appropriate parent and child relationship definitions.

By doing so, any given subsystem can be modeled with optional references to system inputs; by using the respective functions in the genetic programming process, the so modeled a priori knowledge can be incorporated. This procedure might seem to be a bit cumbersome as it would be easier to program functions that have direct access to the data and to use the variables' values directly without needing additional terminals. Still, we have chosen to stick strictly to the original definition of functions as units processing results of other functions or terminals; this is why this approach has been implemented in this manner in HeuristicLab even though there would not have been a technical reason not to provide functions with access to the data basis.

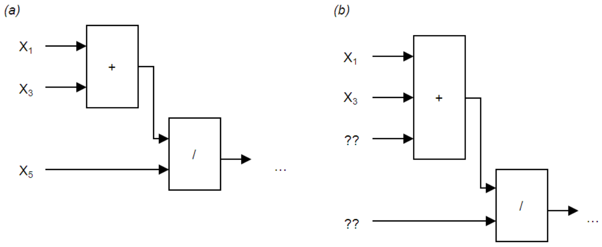


Figure 2: A priori knowledge about the structure of a system.

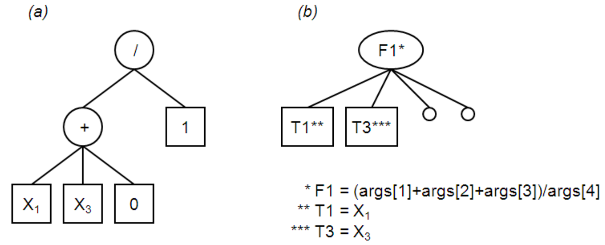


Figure 3: Models representing a priori knowledge given in Figure 2.

4. FORMATION OF NITRIC OXIDES IN DIESEL ENGINES

In fact, research regarding the formulation of dynamic models describing for NO_x emissions of diesel engines has been done since several decades, so there is already a lot of physical and chemical knowledge available for this modeling task:

As Warnatz, Maas and Dibble explain in (Warnatz, Maas, and Dibble 1996), the products of combustion are distinctly identified as a severe source of environmental damage, especially caused by increased combustion of hydrocarbon fuels. The major combustion productions, especially carbon dioxide and water, have long been considered rather "harmless"; now, carbon dioxide is more and more seen as a significant source of problems regarding the atmospheric balance and greenhouse effect. NO_x are less obvious products of combustion; within the last half of the twentieth century it has become apparent that NO and NO_2, collectively called NO_x, are major contributors to photochemical smog and ozone in the troposphere (Seinfeld 1986). Gaining knowledge regarding the production process of NO_x is therefore of great interest and researchers search for models for the production of these pollutants in order to find new ways how to minimize them (Warnatz, Maas, and Dibble 1996).

Based on physical models summarized in (Warnatz, Maas, and Dibble 1996) we have defined the following model describing the production of NO_x depending on measurable engine parameters:

$$MAF^* = \frac{MAF}{N} \cdot \frac{1000}{6} \left[\frac{kg/_h}{U/_{min}} \cdot \frac{1000}{60} = \frac{g}{u} \right] \tag{2}$$

$$NO_x = e^{(qMI \cdot (\alpha \cdot pMI + \beta \cdot \frac{1}{N} + \gamma \cdot MAF^*))} \tag{3}$$

where

- MAF is the amount of fresh air in the engine's intake manifold (MAF here stands for manifold air flow),
- N the engine speed,
- MAF* is the amount of fresh air divided by the rotational frequency and converted to the amount of air per combustion cycle,
- qMI the injected fuel mass per cycle,
- pMI is the crankshaft angle φ before the top dead centre of the piston where fuel injection starts, and
- α, β and γ are parameters which have to be identified.

Figure 4 shows a graphical representation of this semi-abstract model structure:

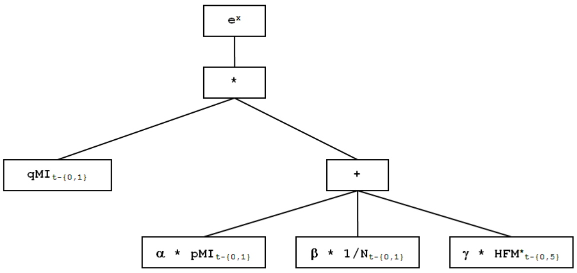


Figure 4: Model representing the physical knowledge available for the formation of NO_x emissions.

5. EMPIRICAL TESTS

The data set we have used for testing the enhanced GP strategies described in the previous sections contains the measurements taken from a 2 liter 4 cylinder BMW diesel engine at a dynamical test bench (simulated vehicle: BMW 320d Sedan). The mean engine speed was set to 2,200 revolutions per minute, and in each engine cycle 15mg fuel were injected. Several emissions (including NO_x, CO and CO_2) as well as several other engine parameters were recorded over approximately 18.3 minutes at 100 Hz and then downsampled to 10 Hz, yielding a data set containing approximately 11,000 samples. 40 signals were recorded, but only 9 variables were considered by the identification algorithm. The reason for this is that information about other emissions should not be incorporated in the model because of redundancies and relatively high costs of exhaust sensors – an emission model using other emission measurements is much easier to be found, but not very significant. Therefore we have only used parameters which are directly measured from the engine's control unit and not in any sense connected to emissions (as for example oil temperature, air pressure, injection parameters etc.).

As we now know about the physical knowledge available in context with formation of NO_x during combustion in diesel engines, there are several ways how we can make this information available for the GP process.

First and most obviously, Formula (2) describing the calculation of the auxiliary variable MAF* can be used for defining a new variable; as there are no parameters to be fixed, this new variable can be introduced into the data base immediately. The GP process is therefore able to use this information simply by using this new variable, i.e. by creating models that reference the variable MAF*.

The incorporation of the information given in Formula (3) is a bit more complicated as it includes parameters which are not known. The first possibility is to seed the population using a stub of the model already known. Of course, this brings along the problem that the unknown parameters included in the model, namely α, β and γ, have to be initialized using some arbitrary, but fixed values; we initially set those parameters to 0.1 and expect the evolutionary optimization process to tune the values so that improved model structures are evolved.

An alternative method is to create an artificial function that represents the structure of the knowledge available. So we define the following additional items that are to be added to the functional basis used by the GP process: The function definition "PKfuncNOx" represents the main part of the model. On the one hand it expects the input variables qMI, pMI, 1/N (N^{-1}) and MAF* as inputs at indices 0, 2, 4 and 6; on the other hand it also expects 5 more inputs that are processed as coefficients (at indices 1, 3, 5 and 8) or an additional term at index 7. When called with the expected inputs $I_{0...8}$, this function returns the result of the expression defined in (4) and graphically displayed in Figure 5:

$$e^{I_0\cdot(I_1\cdot I_2+I_3\cdot I_4+I_5\cdot I_6+I_7)\cdot I_8} \qquad (4)$$

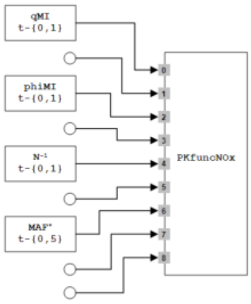


Figure 5: Terminal definitions and the "PKfuncNOx" function representing physical knowledge about the formation of NO_x emissions.

Table 1: Test strategies for incorporating physical knowledge about the formation of NO_x in the GP process.

Index	Parameters
I	No additional information.
II	Use of additional variable MAF*,
III	Use of additional variable MAF* as well as stub model
IIIa	Seed model in 20% of the initial population
IIIb	Seed model in 60% of the initial population
IIIc	Seed model in 100% of the initial population
IV	Use of additional variable MAF* as well as stub model
IVa	Seed model in 10% of the initial population and with 10% probability in main loop
IVb	Seed model in 20% of the initial population and with 20% probability in main loop
IVc	Seed model in 50% of the initial population and with 30% probability in main loop
V	Use of additional variable MAF* and "PKfuncNOx" function
Va	No manipulation of the GP-process
Vb	Introduction of "PKfuncNOx" function into solutions by special mutation operator; probability: 15%

We have applied all strategies presented in Section 3 for incorporating knowledge about the formation of NO_x. In Table 1 we summarize the test strategies actually applied; in all cases we used GP including strict offspring selection (Affenzeller, Wagner, and Winkler 2005), single-point crossover and 12% mutation.

All test strategies were executed 5 times independently using the first 5,500 samples as training data, 3000 samples as validation and the remaining samples as test data. In all test runs we collected those models that performed best on validation data and evaluated them on test data; the results of these test evaluations are summarized in Table 2 where we give statistics about the mean squared errors on test data. Figure 6 illustrates a model which was returned as best model with respect to fit on validation data in one of the test runs in series IVb. Obviously, the given model structures that were inducted into the GP processes in series IV have been used and are incorporated in the model's structure. Figure 7 shows the evaluation of this model on the whole data set.

In (Winkler 2008) the reader can find much more details about the evaluation of these test runs. For example, extensive investigations regarding population dynamics can be found in this thesis as well as further background about test bench setup and the data set used.

Table 2: Quality of results (average and standard deviation), evaluation on test data.

Strategy	Test quality	
	Mean value	Standard deviation
I	0.004359	0.00033
II	0.003901	0.00037
III		
IIIa	0.00388	0.00022
IIIb	0.00408	0.00038
IIIc	0.00429	0.00031
IV		
IVa	0.00382	0.00032
IVb	0.00348	0.00020
IVc	0.00384	0.00027
V		
Va	0.00409	0.00032
Vb	0.00388	0.00028

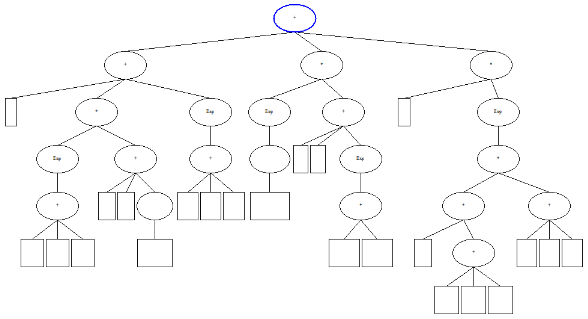


Figure 6: Best model produced for the NO_x data set: The given a priori knowledge has been incorporated as sub-trees of the returned model structure.

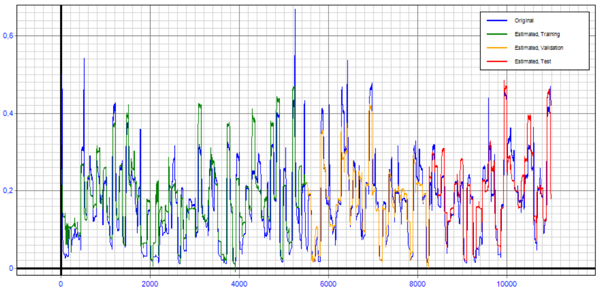


Figure 7: Evaluation of the finally retrieved model.

6. CONCLUSION

In this paper we have summarized test results for an example for the introduction of a priori about the system which is to be identified: Virtual sensors for the NO_x emissions of a BMW diesel engine have been created using physical knowledge. Three different ways how to introduce additional knowledge into the GP based learning process have been discussed and tested:

- If partial knowledge can be formulated by equations without variable parameters, then additional variables can be formed. Of course, this approach can be used in any machine learning approach; in this example application, the virtual variable MAF* has been formed and added to the problem data set, leading to increased model qualities.
- Alternatively, model structures representing partial knowledge can also be introduced into the GP process by seeding parts of the initial population or by repeatedly inserting them into the main GP loop (before crossover and mutation operations). In our example application, this was also successfully done; of course, if this forceful introduction is done too often, then population diversity can be lost leading to worse results.
- The third possibility discussed and tested here is the formation of complex functions representing partial knowledge; the genetic process is then supposed to form models that include these functions. Unfortunately, exactly this approach did not really work fine in this exemplary application: On the one hand, without manipulating the GP process, the function designed in this example died off almost completely, and on the other hand forceful introduction of this function into the existing models had negative effects on population diversity as well as on results quality.

ACKNOWLEDGMENTS

The work described here was done within the research project L284-N04 "GP-Based Techniques for the Design of Virtual Sensors" sponsored by the Austrian Science Fund (FWF). Furthermore, this work was supported by the Linz Center of Competence in Mechatronics (LCM) under grant LCM-001.

REFERENCES

Affenzeller, M., Wagner, S., and Winkler, S., 2005. Goal-oriented preservation of essential genetic information by offspring selection. *Proceedings of the Genetic and Evolutionary Computation Conference 2005*, pp. 1595–1596. June 25-29, 2005, Washington, D.C., USA.

Alberer, D., del Re, L., Winkler, S., and Langthaler, P., 2005. Virtual sensor design of particulate and nitric oxide emissions in a DI diesel engine. *Proceedings of the 7th International Conference on Engines for*

Automobile ICE 2005, paper number 2005-24-063. September 11-16, 2005, Capri, Italy.

del Re, L., Langthaler, P., Furtmüller, C., Winkler, S. and Affenzeller, M., 2005. NO_x Virtual Sensor Based on Structure Identification and Global Optimization. *Proceedings of the SAE World Congress 2005*, paper number 2005-01-0050. April 11-14, 2005, Detroit, MI, USA.

Koza, J., 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: The MIT Press.

Keijzer, M. and Babovic, V., 1999. Dimensionally aware genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, pp. 1069–1076. July 13-17, 2005, Orlando, FL, USA.

Langdon, W.B. and Poli, R., 2002. *Foundations of Genetic Programming*. Berlin, Germany: Springer Verlag.

Ljung, L., 1999. *System Identification - Theory For the User*, 2nd edition. Upper Saddle River, NJ, USA: PTR Prentice Hall.

Seinfeld, J.H., 1986. *Atmospheric Chemistry and Physics of Air Pollution*. Somerset, NJ, USA: John Wiley and Sons, Inc.

Wagner, S. and Affenzeller, M., 2005. HeuristicLab: A generic and extensible optimization environment. *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms*, pp. 538-541. March 21-23, 2005, Coimbra, Portugal.

Wagner, S. and Affenzeller, M., 2005. SexualGA: Gender-Specific Selection for Genetic Algorithms. *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics 2005*, pp. 76-81. July 10-13, 2005, Orlando, FL, USA.

Warnatz, J., Maas, U. and Dibble, R.W., 1996. *Combustion - Physical and Chemical Fundamentals, Modeling and Simulation, Experiments, Pollutant Formation*. Berlin, Germany: Springer Verlag.

Winkler, S., Affenzeller, M. and Wagner, S., 2006. HeuristicModeler: A Multi-Purpose Evolutionary Machine Learning Algorithm and its Applications in Medical Data Analysis. *Proceedings of the International Mediterranean Modelling Multiconference I3M 2006*, pp. 629-634. October 4-6, 2006, Barcelona, Spain.

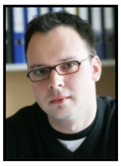
Winkler, S., Affenzeller, M. and Wagner, S., 2007. Advanced Genetic Programming Based Machine Learning. *Journal of Mathematical Modelling and Algorithms*, 6: 455-480.

Winkler, S., Affenzeller, M. and Wagner, S., 2007. Selection Pressure Driven Sliding Window Genetic Programming. *Computer Aided Systems Theory - EUROCAST 2007, Lecture Notes in Computer Science* 4739: 788-795.

Winkler, S., 2008. *Evolutionary System Identification - Modern Concepts and Practical Applications*. PhD Thesis, Johannes Kepler University Linz. April 2008, Linz, Austria.

AUTHORS BIOGRAPHIES

STEPHAN M. WINKLER received his MSc in computer science in 2004 and his PhD in engineering sciences in 2008, both from JKU Linz, Austria. His research interests include genetic programming, nonlinear model identification and machine learning. Currently he is research associate at the Research Center Hagenberg of the Upper Austrian University of Applied Sciences, working on the research program L284-N04 "GP-Based Techniques for the Design of Virtual Sensors", a research project funded by the Austrian Science Fund (FWF).



MARKUS HIRSCH received his MSc in Mechatronics from JKU Linz, Austria in 2006; since then he has been research associate at the Linz Center of Competence in Mechatronics (LCM), also within the research program L284-N04 "GP-Based Techniques for the Design of Virtual Sensors". His research interests include system identification, nonlinear modeling, design of experiment, biological systems, automotive control, and engine control and optimization.

MICHAEL AFFENZELLER has published several papers and journal articles dealing with theoretical aspects of evolutionary computation and genetic algorithms. In 1997 he received his MSc in mathematics and in 2001 his PhD in engineering sciences, both from JKU Linz, Austria. He is professor at the Upper Austria University of Applied Sciences (Campus Hagenberg) and associate professor at the Institute of Formal Models and Verification at JKU Linz since his habilitation in 2004.

LUIGI DEL RE is professor at Johannes Kepler University Linz and head of the Institute for Design and Control of Mechatronical Systems. He received his MSc in electronic engineering in 1981 and his PhD degree in technical sciences in 1990, both from ETH Zürich, Switzerland. His research interests include nonlinear control, parameter and structure identification, automotive control, model predictive control, and systematic design assessment.

STEFAN WAGNER also received his MSc in computer science in 2004 from Johannes Kepler University Linz, Austria. He currently holds the position of an associate professor at the Upper Austrian University of Applied Sciences (Campus Hagenberg). His research interests include evolutionary computation and heuristic optimization, theory and application of genetic algorithms, machine learning and software development.